Merging Flows in Terminal Moneuvering Area using Time Decomposition Approach

Ji Ma, Daniel Delahaye, Mohammed Sbihi, Marcel Mongeau

Merging Flows in Terminal Maneuvering Area using Time Decomposition Approach

Ji Ma, Daniel Delahaye, Mohammed Sbihi, Marcel Mongeau
ENAC – Université de Toulouse
7 av. Edouard Belin, 31055 Toulouse cedex 4, France

***Abstract*—With a continuous growth of air traffic demand, more effort must be made to alleviate the current overloaded airspace charges. This research focuses on the aircraft merging and sequencing problem at Terminal Maneuvering Area. Tactical conflict detection and resolution methods are applied to a predefined route network structure. Speed and time changes are proposed via an optimization methodology to resolve conflicts and maintain separation between aircraft with regard to the wake turbulence constraints and runway occupancy time. A new time decomposition approach is introduced. It consists in partitioning the whole time interval under consideration into several overlapping time windows, and in solving the merging and sequencing problem individually in each such sub-window. Four aircraft status are defined to classify flights according to their temporal position relative to the current sliding window. Moreover, an adapted simulated annealing heuristic is proposed to solve the corresponding sub-problems. Finally, computational experiments of the proposed algorithm, performed on real-world case studies of Paris Charles De-Gaulle airport, show the benefits of this sliding-window time-decomposition approach.**

Keywords-Air Traffic Control; Aircraft Merging Problem; Time Decomposition; Simulated Annealing

I. Introduction

The global air traffic will grow at 4.6 percent annually for the next 20 years according to the Airbus global market forecast [1]. Therefore, the research on Air Traffic Management (ATM) in order to ensure safety and to enhance efficiency and capacity has become more and more important. One of the most critical parts related to ATM is the Terminal Maneuvering Area (TMA). The TMA is a designated area of controlled airspace surrounding a major airport where there is a high volume of traffic. This complex airspace has become a sensitive area with an uncertain and dynamic environment. Therefore, there is a need to improve the use of available air capacity and to alleviate the more and more serious airspace congestion by using efficient approaches and algorithms. In this paper, we focus on the problem of merging flight flows into the TMA in order to ensure safety and improve efficiency of the TMA.

The remainder of this paper is organized as follows. First, the background and related research, the precise description of the problem including the objectives and constraints, and a mathematical model are introduced in Section II. Then, the resolution algorithms for the merging and sequencing problem are proposed in Section III. Next, we perform experiments and we analyze the results in Section IV. Finally, Section V gives some conclusions and perspectives.

II. Problem description and mathematical model

A. Problem description

During the transition from the en-route to the terminal airspaces, aircraft arriving from different entry points must be merged and organized into an orderly stream, typically in a short time horizon of about 45 minutes [2].

The aircraft arrival sequencing and scheduling problem (ASS) has been considered as a classical problem for the last few decades. Beasley *et al.* [3] present mixed-integer zero-one programs of the static aircraft landing problem for both the single and multiple-runway cases. Bianco *et al.* [4] proposed a job-shop scheduling model combining a local search algorithm for the ASS problem with multiple runways and multiple approach and leaving procedures. An overview of different aircraft landing and take-off scheduling techniques is summarized in [5] where the main research works, including heuristics, meta-heuristics, branch and bound, and dynamic programming, are presented.

The First-Come-First-Served (FCFS) order is the most common technique that controllers use to sequence aircraft. Although this approach reasonable to maintain equity of scheduling, FCFS does not consider most useful criteria to alleviate congestion and does not make an efficient use of the airport capacity, due to its large spacing requirement. Typically, controllers may shift aircraft by a small number of rank positions in the FCFS order, which means that each aircraft must land within a pre-specified number of positions of its place in the FCFS sequence. This explains why constrained position shifting (CPS) methods have been widely studied. Balakrishnan et al. [6] present a Dynamic Programming approach subject to several operational constraints in a CPS environment to minimize the *makespan* (landing time of the last aircraft).

Some researches have been conducted by decomposing the problem into a sequence of sub-problems based on a sliding time window. Hu *et al.* [7] design an efficient Genetic Algorithm based on a binary representation of arriving queues combined with a receding horizon control. Furini *et al.* [8] propose an improved rolling-horizon approach which partitions the aircraft sequence into *chunks* (sub-sequences), and solves the corresponding sub-problems individually with a tabu-search heuristic.

In this paper we consider the problem of merging arrival flows on a single runway using a time decomposition

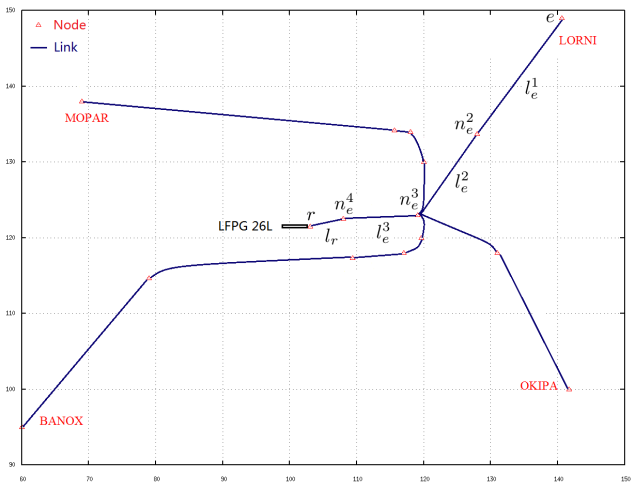
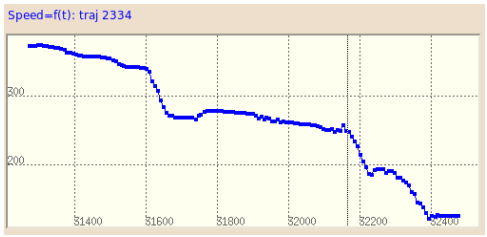
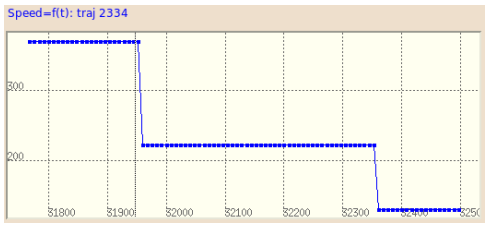


Fig. 1. The route structure model for LFPG runway 26L



(a) Real aircraft speed profile with respect to time



(b) Speed change model

Fig. 2. Example of real aircraft speed profile and speed change model with respect to time

approach. Next, a mathematical model for this problem is introduced. It is the work of Zuñiga *et al.* [2] which aims to remove conflicts at merging points and to maintain separation between aircraft following same route link. In their work, a genetic algorithm is applied to Gran Canaria airport in Spain with a one-hour real data set.

B. Given data

We assume that a route network graph $G(\mathscr{N}, \mathscr{L})$, in which the aircraft are allowed to fly, is given, where $\mathscr{N}$ is the node set and $\mathscr{L}$ is the link set. We have $\mathscr{N} = N_e \cup N_w \cup \{r\}$, where $N_e = \{1, \ldots, n_e\}$ represents the set of entering points, n_e is the total number of entering points, r represents the node at the runway threshold, and N_w is the set of nodes between the entering point until the runway threshold which constitutes the route network. Similarly, we have $\mathscr{L} = L_e \cup L_w \cup \{l_r\}$, where $L_e = \{1, \ldots, n_e\}$ is the set of the links connecting the entering point, l_r is the last link connecting the runway threshold, and L_w contains the remaining links. Each entering point $e \in N_e$ corresponds to one route, defined by $r_e = \{e, l_e^1, n_e^2, l_e^2, \ldots, n_e^{r-1}, l_r, r\}$. Each route is defined by a succession of nodes and links; the first link starts from the entering point, e, and the last link ends at the runway, r. The set of routes is denoted as $\mathscr{R} = \{r_e\}_{e=1}^{n_e}$. Each aircraft follows exactly one of these routes corresponding to its entering point.

Fig. 1 displays a model example of a route network: Paris Charles De-Gaulle (CDG) airport runway 26L. In the arrival procedure, four routes fuse into one single route towards the runway. Each of the starting nodes of these four routes is a so-called *Initial Approach Fix* (IAF). The set of entering points is N_e={MOPAR, LORNI, OKIPA, BANOX}, with $n_e = 4$. In this route network structure, there are $|\mathscr{N}| = 16$ nodes and $|\mathscr{L}| = 15$ links.

Assume that we are given a set of flights (or aircraft), $\mathscr{F} = \{1, \ldots, N_f\}$, and for each flight $f \in \mathscr{F}$ the following data is also given:

- $e_f \in N_e$: entering waypoint number at TMA;
- t_s^f: RTA (Required Time of Arrival) at TMA;
- v_s^f: entering speed at TMA;
- c_f: wake turbulence category.

Let us now detail the assumptions and simplifications required to address our problem.

First, the aircraft speed is considered to be equal to the Ground Speed (the wind effect will not be taken into account). Next, in order to understand our speed profile assumption, consider the following example: the real-world operations of an aircraft of type B737 that lands at CDG airport. First, the ICAO rules require the maximum speed not to exceed 250 kt below FL100, which corresponds to a distance of 30 NM from the runway threshold. Moreover, this aircraft must reduce its speed to 250 kt at a distance of 11 NM from the runway threshold, and to 140 kt at a distance of 3 NM from the runway threshold. The aircraft is assumed to reduce its speed in a way that is compatible with the above-mentioned rules. Figure 2a shows an example of how an aircraft changes its speed during the final approach. In our model, in order to remain close to the reality while simplifying the problem to optimize, we approximate the real speed profile as a step function, as illustrated in Fig. 2b. More precisely, we assume that the initial speed of each aircraft is kept constant throughout the first link. Then, from the beginning of the second link of the route, the aircraft speed is decreased so as to meet the ICAO requirements. Next, the aircraft keeps this constant speed for the remaining of the route, except for the last link. When the aircraft arrives at the beginning of its last link, the speed is decreased again, to reach the final-approach speed, noted as v_f^r. This final-approach speed depends on c_f, as follows:

$$v_f^r = \begin{cases} 150\ kt, & \text{if } c_f = \textit{Heavy} \\ 130\ kt, & \text{if } c_f = \textit{Medium} \\ 110\ kt, & \text{if } c_f = \textit{Small} \end{cases} \quad (1)$$

and is kept constant throughout the last link of the route.

C. Decision variables

We consider two possible maneuvers for each flight to resolve the potential conflicts:

- Shifting its entering time at TMA;
- Modifying its entering speed.

First, we assume that we are given a maximum delay and a minimum delay, denoted respectively $\Delta t_{\max}$ and $\Delta t_{\min}$, which define the range of possible entering times at TMA. We choose to discretize this time interval because in the real ATC environment, the controller radars usually display updates every five seconds and discretizing the time-slot variable makes more sense for ATC practice. We therefore define, for each flight $f \in \mathscr{F}$, a time-slot decision variable $t_f \in T_f$, where

$$T_f = \{t_s^f + j\Delta t \ | \Delta t_{\min}/\Delta t \leqslant j \leqslant \Delta t_{\max}/\Delta t, \ j \in \mathbb{Z}\},$$

where Δt is a discretized time increment, an input parameter whose value is to be set by the user. In order to shift an aircraft entering time at TMA, we can either delay it or speed it up during the en-route procedure. In practice, the latter strategy consumes more fuel, and may be far less interesting for the airlines. As a consequence, our time slot interval is asymmetric, with $|\Delta t_{\max}| \geq |\Delta t_{\min}|$. In this study, we set $\Delta t_{\max} = 500s$, $\Delta t_{\min} = -100s$ and $\Delta t = 5s$.

The second decision of our conflict resolution strategy consists in changing the entering speed of the flights. Again, we choose to discretize the interval of possible speed values, in accordance with current practice of controllers, who usually modify aircraft speed by accelerating or reducing by a discretized value. Thus, for each flight $f \in \mathscr{F}$, we define an entering speed decision variable $v_f \in V_f$, where

$$V_f = \{v_{\min}^f + j\Delta_v^f \ | \ j \in \mathbb{Z}, |j| \leqslant (v_{\max}^f - v_{\min}^f)/\Delta_v^f\},$$

where Δ_v^f is a (user-defined) time increment, $v_{\min}^f$ and $v_{\max}^f$ are given input data corresponding to the minimum and maximum allowable speeds for aircraft f. In this study, we set $v_{\min}^f = 0.9v_s^f$, $v_{\max}^f = 1.1v_s^f$ and $\Delta_v^f = 0.01v_s^f$. We also make sure *a priori* that $v_{\max}^f$ is not exceeding the maximum speed defined by the aircraft type.

As mentioned before, our speed-decrease model follows a step-function profile, as illustrated in Fig. 2b. For the first link of aircraft f, the speed remains constant at v_f. Then, from the second link until the beginning of the last link, it is reduced to a constant speed αv_f, where α is a parameter whose value is set to:

$$\alpha = \begin{cases} 1.0, & \text{if } v_f \leq 250\text{kt} \\ 0.7, & \text{if } 250\text{kt} < v_f \leq 360\text{kt} \\ 0.6, & \text{if } v_f > 360\text{kt} \end{cases} \quad (2)$$

Finally, for the last link, the aircraft keeps its speed at the constant value given by (1) until the runway.

To summarize, our decision vector is $\mathbf{x} = (\mathbf{t}, \mathbf{v})$, where $\mathbf{t}$ is the vector whose i^{th} component is the decision variable t_i, and $\mathbf{v}$ is the vector whose i^{th} component is the decision variable v_i (both of which correspond to flight i).

D. Separation rules

We consider three separation requirements: wake turbulence constraints, horizontal separation, and runway separation.

- **Wake turbulence separation constraints**:
  In order to model the wake turbulence constraints, let us define s_{fg}, the minimum allowed separation between flights f and g, a given input data that depends on the wake turbulence categories, c_f, c_g, of these two flights. The minimum separation standards are given in Table I [9].

TABLE I
SEPARATION MINIMA, s_{fg}, IN NM.

Category		**Leading Aircraft,** f		
		Heavy	***Medium***	***Light***
Trailing Aircraft, g	***Heavy***	4	3	3
	Medium	5	3	3
	Light	6	5	3

- **Horizontal separation constraints:** Aircraft must satisfy a minimum horizontal separation based on radar, which shall be 3 NM in the TMA.
- **Runway separation constraints:** The runway separation rules are calculated by incorporating the different flight velocities and their impact on the final approach segment. Here we use the data of [10], shown in Table II:

TABLE II
SINGLE-RUNWAY SEPARATION REQUIREMENTS, IN SECONDS.

Category		**Leading Aircraft,** f		
		Heavy	***Medium***	***Light***
Trailing Aircraft, g	***Heavy***	96	60	60
	Medium	157	69	69
	Light	207	123	82

E. Conflicts detection

We explain here how to evaluate the above-defined separation constraint functions. Three kinds of conflicts are defined: *link conflict*, *node conflict* and *runway conflict*, in order to meet the requirements of wake turbulence separation, horizontal separation and runway separation respectively. Note that once the decision variable values are set, we can calculate the corresponding times at which the aircraft passes each node and each link. Then, we use these time values to determine the three types of conflicts. These are defined below.

- **Link conflict:** For each given link, we verify twice whether a conflict occurs, *i.e.*, the minimum wake turbulence separation is violated: at the entry and at the exit of the link. Moreover, we make sure that the order of sequencing remains the same along the link. As the speed is assumed to stay constant when the aircraft flies through one link, if no conflict is detected at the entry nor at the exit of the link, then this link is guaranteed to be conflict free. Fig. 3 illustrates an example of link conflict detection: two consecutive flights f, g are flying

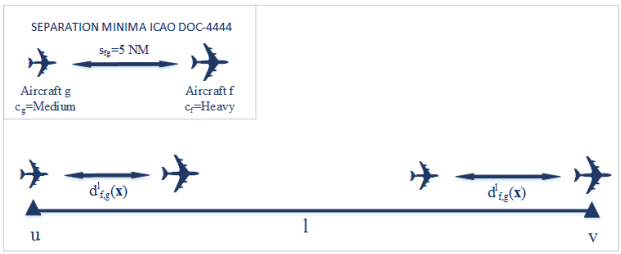


Fig. 3. Example of link conflict detection

through a link $l = (u, v)$. The minimum separation between these two aircraft, s_{fg}, is obtained based on their respective wake turbulence category (Table I). Then, the actual separation distance of these aircraft at the entry time, $d^u_{fg}(\mathbf{x})$, and at the exit time of link l, $d^v_{fg}(\mathbf{x})$, are computed and compared with s_{fg} (here $s_{fg} = 5$NM) to detect an eventual link conflict.

Let us define, the *link conflict indicator*, $L^l_{fg}(\mathbf{x})$, for aircraft f and g at link l:

$$L^l_{fg}(\mathbf{x}) = \begin{cases} 1, & \text{if } T^u_f(\mathbf{x}) < T^u_g(\mathbf{x}) \text{ and} \\ & (d^u_{fg}(\mathbf{x}) < s_{fg} \\ & \text{or } d^v_{fg}(\mathbf{x}) < s_{fg} \\ & \text{or } T^v_f(\mathbf{x}) > T^v_g(\mathbf{x})) \\ 0, & \text{otherwise} \end{cases}$$

- **Node conflict:** If no link conflict is detected, wake-turbulence separation can be guaranteed. However, at the intersection of two successive links, violation of the horizontal separation requirement between two consecutive aircraft can still occur. Therefore, we introduce the node conflict, illustrated in Fig. 4. Considering a node n and two aircraft f, g that fly over node n, we consider a disk centered at node n with radius R_n, defined as a *detection zone*. We must ensure that at every moment only one aircraft passes this detection zone. Suppose that aircraft f enters the zone of node n before aircraft g. We denote the entering time to and exit time from this zone for aircraft f (g, respectively) as $T^{f,n}_{\text{In}}(\mathbf{x})$ ($T^{g,n}_{\text{In}}(\mathbf{x})$) and $T^{f,n}_{\text{Out}}(\mathbf{x})$ ($T^{g,n}_{\text{Out}}(\mathbf{x})$). A conflict is detected when $T^{g,n}_{\text{In}}(\mathbf{x}) < T^{f,n}_{\text{Out}}(\mathbf{x})$, which means that aircraft g enters the detection zone before aircraft f exits.

  Zuñiga *et al.* [2] choose $R_n = 3$NM, which ensures the minimum required separation. However, it induces some waste of separation and may limit the possibility of finding a good-quality solution. Therefore, we propose a value of R_n that takes into account the worst case of aircraft speeds and intersection angles based on the route map of CDG runway 26L. Suppose that the following aircraft speed, v_g, is higher than the leading aircraft speed, v_f. They pass consecutively the node n, which is the intersection of two perpendicular links. In our route network, the minimum angle between two consecutive links of one route is larger than 90 degrees. Therefore, the worst case to consider for our node conflict detection

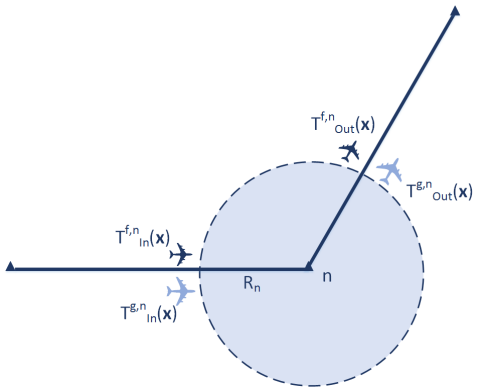


Fig. 4. Example of node conflict detection

is the perpendicular link illustrated on Fig. 5. We suppose that the node n is the origin point of our coordinate axes, and that at time 0 flight f arrives at node n. To ensure that aircraft f exits the zone of node n before aircraft g enters, the minimum distance between the two aircraft when the first one is at node n should be $d_{fg}(0) = (1+v_g/v_f)R_n$. The coordinates of flights f and g as a function of the time t are:

$$\begin{cases} x_f(0) = 0 \\ x_f(t) = v_f t \\ y_f(t) = 0 \end{cases} \tag{3}$$

$$\begin{cases} y_g(0) = (1 + v_g/v_f)R_n \\ x_g(t) = 0 \\ y_g(t) = y_g(0) - v_g t \end{cases} \tag{4}$$

and the distance between them at time t is:

$$d_{fg}(t) = \sqrt{(x_g(t) - x_f(t))^2 + (y_g(t) - y_f(t))^2} \tag{5}$$

After the deviation, the time at which this is minimal is:

$$t_{min} = \frac{(1 + v_g/v_f)v_n}{v_g^2 + v_f^2} \tag{6}$$

The corresponding distance, is:

$$d_{fg}(t_{min}) = \frac{(v_g + v_f)R_n}{\sqrt{v_g^2 + v_f^2}}, \tag{7}$$

should be larger than 3NM. We consider the extreme case of aircraft speed in our problem: $v_f = 250$kt and $v_g = 430$kt, which corresponds to the minimum and maximum speeds in our data. After calculation, we obtain a radius of $R_n = 2.2$NM.

We define the *node conflict indicator* for aircraft f (leading) and g (following) as follows:

$$N^n_{fg}(\mathbf{x}) = \begin{cases} 1, & \text{if } T^{f,n}_{\text{In}}(\mathbf{x}) \leq T^{g,n}_{\text{In}}(\mathbf{x}) < T^{f,n}_{\text{Out}}(\mathbf{x}) \\ 0, & \text{otherwise} \end{cases}$$

Moreover, as the angle between two links becomes small,

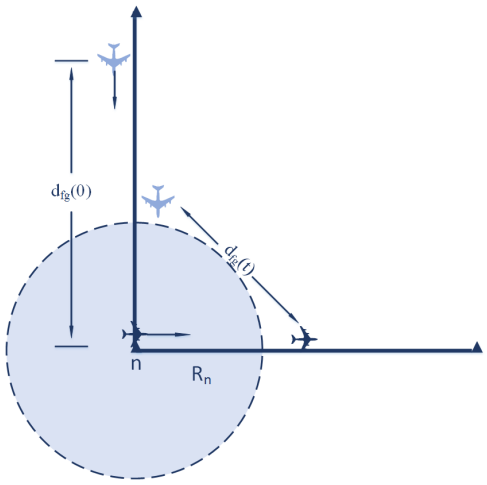


Fig. 5. Aircraft minimum distance calculation for node conflict

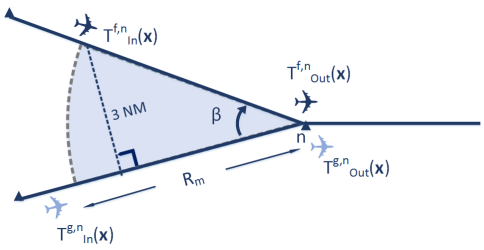


Fig. 6. Example of merge conflict detection

undetected conflict outside the detection zone are likely to occur without being detected. In order to address this issue, we redefine and adapt to the conflict detection zone depending on the angle between two links. When two aircraft f and g come from different directions and merge at one node n, a fan-shaped area with a radius R_m is defined as the detection zone, as illustrated in Fig. 6, where $R_m = 3/sin\beta$ NM, with β calculated by using the coordinates of the three nodes that constitute this angle. Similarly to the node conflict detection, suppose that aircraft f exits the fan zone (detection zone) of node n before aircraft g enters. We note the entering and the exit times, and we ensure that aircraft g enters the node zone before aircraft f exits.

- **Runway conflict:** The landing time difference of any two consecutive aircraft must respect the time separation that we mentioned before, considering the different speeds and runway occupancy times. Otherwise, a *runway conflict* is incurred. We define the *runway conflict indicator* between two successive aircraft f and g as:

$$P_{fg}(\mathbf{x}) = \begin{cases} 1, & \text{if } 0 \leq T^r_g(\mathbf{x}) - T^r_f(\mathbf{x}) < t_{fg} \\ 0, & \text{otherwise} \end{cases}$$

where $T^r_f(\mathbf{x})$ denotes the time at which aircraft f arrives at the last node r (the runway threshold), which corresponds to its landing time, and t_{fg} is the minimum runway separation between flights f and g.

F. Objective function

The objective is to minimize the total number of conflicts in the nodes, in the links, and at the runway threshold for each flight. At the same time, we aim at minimizing the number of flights that must implement a change in decision. Our objective function, to be minimized is:

$$S(\mathbf{x}) = \lambda \left(\sum_{\substack{f,g \in \mathscr{F} \\ f \neq g}} \left(\sum_{n \in r_f \cap r_g} N^n_{fg}(\mathbf{x}) + \sum_{l \in r_f \cap r_g} L^l_{fg}(\mathbf{x}) + P_{fg}(\mathbf{x}) \right) \right) + \gamma D(\mathbf{x})$$

where $D(\mathbf{x}) = |\{f \in \mathscr{F} | t_f(\mathbf{x}) \neq t^f_s \text{ or } v_f(\mathbf{x}) \neq v^f_s\}|$, is the number of flights implementing a change in decision, the first item represents the total number of conflicts in all the nodes, all the links and at the runway, λ and γ are weighting coefficients for the total number of conflicts and the decision deviations respectively. The user can adjust these coefficients to find a solution resolving all conflicts without implementing too many changes in decision.

This optimization problem is proven to be *nondeterministic polynomial* (NP) *hard* [3], which, roughly speaking, means that the computation times to solve it is likely to increase exponentially as the size of the problem grows. However, in practice, controllers need a quick and efficient (not necessarily optimal) solution. In the next section, a rapid and efficient resolution approach will be proposed.

III. Solution Approaches

We propose a time decomposition approach combined with a simulated annealing algorithm to address the aircraft merging problem, modeled as the conflict minimization problem introduced in the previous section. First, we introduce the new concept of sliding-window approach. Then, a simulated annealing algorithm adapted to this problem and to this sliding-window approach is proposed.

A. Sliding-window approach

The approach we are proposing addresses the original problem by decomposing it into several sub-problems using a sliding window in order to reduce the computational burden. This specific approach is generic and can be extended and applied to other real-time operation problems.

Let us introduce some notations. Suppose that we are given a total time interval, $[t_{\text{INIT}}, t_{\text{FINAL}}]$, over which we want to optimize. Then, four parameters are introduced:

- W: the time length of the sliding window;
- S: time shift of the sliding window at each iteration;
- $T_s(k)$: the starting time of the k^{th} sliding window, $T_s(k) = t_{\text{INIT}} + kS$;
- $T_e(k)$: the ending time of the k^{th} sliding window, $T_e(k) = t_{\text{INIT}} + kS + W$.

Fig. 7 illustrates how the operating window slides along the time axis. The first sliding window begins at t_{INIT} and, the

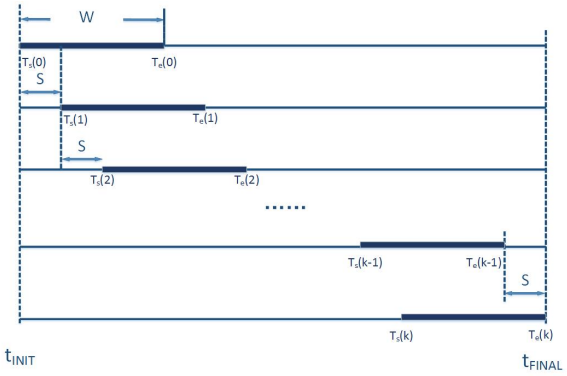


Fig. 7. Sliding windows from iteration 0 to iteration k

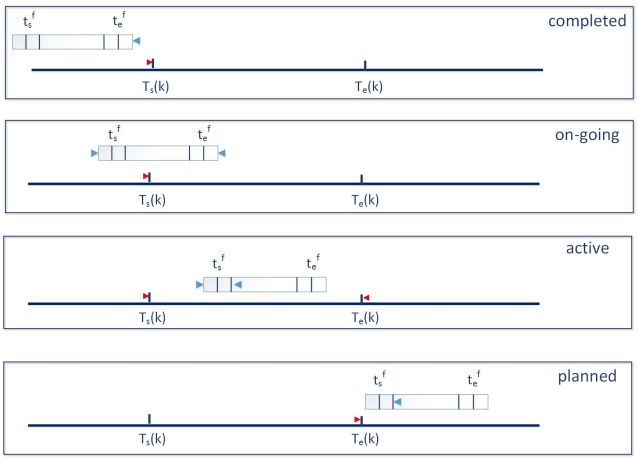


Fig. 8. Four flights status, related to the time position of flight f relative to the current sliding window (k)

optimization algorithm (to be defined later) is applied in the corresponding time interval $[T_s(0), T_e(0)]$. Next, the sliding window recedes in the future by S, and the current optimizing interval becomes $[T_s(1), T_e(1)]$. Then, we repeat this receding process until we reach the k^{th} sliding window with $T_e(k) = t_{\text{FINAL}}$.

Some parameters are needed to describe the sliding-window approach for each flight $f \in \mathscr{F}$:

- t_s^f: initial entering time at TMA;
- $\underline{t_s^f}$: the earliest possible entering time at TMA (initially $\underline{t_s^f} = t_s^f + \Delta t_{\min}$);
- $\overline{t_s^f}$: the latest possible entering time at TMA, (initially $\overline{t_s^f} = t_s^f + \Delta t_{\max}$);
- t_e^f: the landing time;
- $\underline{t_e^f}$: the earliest possible landing time. Initially calculated by considering the maximum allowed speed and the minimum delay;
- $\overline{t_e^f}$: the latest possible landing time. Initially calculated by considering the minimum allowed speed and the maximum delay.

Each aircraft is classified into one of the following four different status, based on the positions of the parameters of flight f relative to the starting and ending times of the current sliding window, k:

- **Completed flight:** $\overline{t_e^f} \leqslant T_s(k)$. The latest landing time for aircraft f, $\overline{t_e^f}$, is lower than the beginning time of the k^{th} sliding window, $T_s(k)$, which means that aircraft f has already landed before the start of the k^{th} sliding window;
- **On-going flight:** $\underline{t_s^f} \leqslant T_s(k) < \overline{t_e^f}$. The beginning time of the k^{th} sliding window, $T_s(k)$, is between the earliest arrival time at TMA, $\underline{t_s^f}$, and the latest landing time, $\overline{t_e^f}$, which means that aircraft f has already been assigned, but it may still impact the assignment of the following aircraft;
- **Active flight:** $T_s(k) < \underline{t_s^f}$ and $\overline{t_s^f} \leqslant T_e(k)$. The time decision interval of flight f is included in the sliding window interval $[T_s(k), T_e(k)]$;
- **Planned flight:** $T_e(k) < \overline{t_s^f}$. The latest arrival time at TMA, $\overline{t_s^f}$, is larger than the ending time of the k^{th} sliding window, $T_e(k)$, which means that the temporal decision variable interval is not totally included in the time window, so that we could not solve the problem in this interval. The flight will be considered later.

The status of flight f is updated and changed according to the sliding window currently under consideration. Remark that for completed and on-going aircraft, we have $\underline{t_s^f} = t_s^f = \overline{t_s^f}$ and $\underline{t_e^f} = t_e^f = \overline{t_e^f}$, since their decision variables have been assigned and fixed.

Fig. 8 illustrates the four different flight status and their positions relative to the sliding window. The different time positions of the aircraft and those of the sliding-window are indicated respectively with blue and red triangles.

At each step, we take into account the active and on-going aircraft in the sliding window interval to be merged and sequenced. Decisions for the on-going flights have already been made, but these flights still have some influence on the decisions to be made for the active flights. On the other hand, the conflicts involving completed flights have already been resolved and they can not have any impact on the active flights, so they can be cleared out of the decision process and ignored. Then, the optimization window recedes in the future by the time step S. The status of aircraft are updated, a new set of flights waiting to be addressed are considered, and the optimization process is repeated, as illustrated in Algorithm 1.

B. Simulated Annealing Algorithm

Simulated Annealing (SA) is a meta-heuristic well known for its ability to trap out of local minima by allowing random local changes. Moreover, it can easily be adapted to large-scale problems with continuous or discrete search spaces.

We propose a SA algorithm adapted to our problem. First, a *neighborhood function* is defined to generate a local change from the current solution. Two criteria are considered: minimizing the computational time and keeping the proposed

Algorithm 1 Sliding-Window Management

procedure SLIDINGWINDOW
  $k \leftarrow 0$;
  $T_s(k) \leftarrow t_{\text{INIT}}$;
  $T_e(k) \leftarrow T_s(k) + W$;
  Determine each flight status relative to sub-window;
  $F_{\text{OPT}} \leftarrow$ Active and on-going flights;
  while $T_e(k) < t_{\text{FINAL}}$ **do**
    if at least one active flight in F_{OPT} **then**
      Subproblem: optimize considering F_{OPT};
    end if
    $T_s(k) \leftarrow T_s(k) + S$;
    $T_e(k) \leftarrow T_e(k) + S$;
    $k \leftarrow k + 1$;
    Update each flight status relative to sub-window;
    Update F_{OPT};
  end while
end procedure

changes local, so that each step does not boil down to a pure random search.

To generate a neighborhood solution, instead of simply choosing randomly a flight f in the active-flight set, we use a method similar to the *roulette wheel* selection. Let us denote C_f the total number of conflicts involving aircraft f. Suppose that we have a total of N active flights, indexed in chronological order (with respect to our entering times) in a sliding window. We propose to change the decision variables of the first aircraft i satisfying $\sum_{f=1}^{i} C_f \geq \mu \sum_{f=1}^{N} C_f$, where μ is a random number between 0 and 1.

The fact that our neighborhood definition is based on the total number of conflicts, augments the likelihood that a flight involving many conflicts, or its neighboring aircraft, will be chosen.

At the level of modifying the values of the decision variables related to the chosen flight f, two strategies are applied: modifying v_f, the entering speed at TMA, or modifying t_f, its entering time at TMA. Then, the new flight trajectory information is updated with regard to the modified decision variables.

IV. Simulation Results

This section compares two resolution approaches to solve the merging and sequencing problem. Numerical results with different settings of (user-defined) algorithm parameters are presented and discussed.

The study is based on three 24-hour real data cases of arrival traffic at Paris CDG Airport on runway 26L. The overall process is run on a 2.33GHz Debian 3.2 Linux operating system PC based on a Java code.

A. Real data analysis

We choose the data set of three consecutive days to test our algorithm viability and performance. The data of November 24, 2015, November 25, 2015 and November 26, 2015 are noted as scenario 1, scenario 2 and scenario 3 respectively. The daily traffic flow characteristics and the repartition are presented in Table III and in Fig. 9 respectively.

TABLE III
DAILY TRAFFIC FLOW CHARACTERISTICS

Scenario	Arrivals	Entry Node	Medium	Heavy	Traffic flow proportion
1	239	MOPAR	23	14	15.5 %
		LORNI	57	22	33 %
		OKIPA	72	6	31.8 %
		BANOX	44	3	19.7 %
2	355	MOPAR	40	22	17.5 %
		LORNI	72	24	27 %
		OKIPA	101	25	35.5 %
		BANOX	65	6	20 %
3	374	MOPAR	37	30	17.9 %
		LORNI	66	31	25.9 %
		OKIPA	117	18	36.1 %
		BANOX	65	10	20.1 %

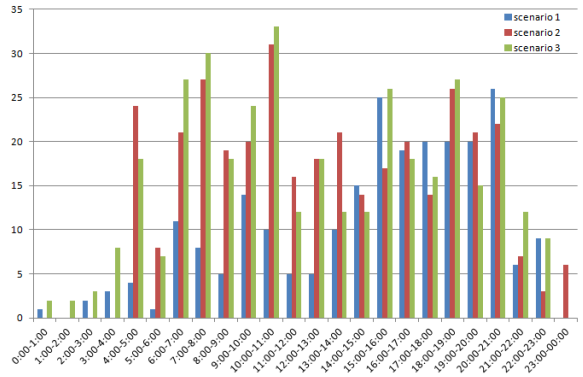


Fig. 9. Traffic flow of Paris CDG runway 26L on three days traffic

B. Parameters setting and results comparison

To evaluate the benefit of our sliding-window approach, a comparative experiment is performed, involving applying our SA algorithm implementation at once on the 24-hour data, and combing SA and the sliding-window approach.

The related user-defined parameters for the SA algorithm and the sliding-window approach are set empirically to the values shown in Table IV and kept constant for both the pure SA algorithm and the SA algorithm within sliding-window approach, and for the three scenarios. The deviation coefficient, γ, is set to a much lower value than λ, in order to ensure that the conflicts resolution is the first priority.

Table V shows the potential (initial) conflicts for the three scenarios without any conflict-resolution strategy applied. We reach a conflict-free solution for scenario 1 and 2 with both approaches. As expected, the sliding-window approach requires less CPU time for the three scenarios, as it addresses smaller NP-hard problems. In scenario 3, there remain unsolved conflicts. However, the sliding-window approach enables us to decrease the number of unsolved conflicts from 48 (for the pure SA approach) down to 14. In this preliminary study, each

TABLE IV
USER-DEFINED PARAMETER VALUES

Parameter	Value
Number of iterations at each temperature step	150
geometrical temperature-reduction coefficient	0.95
Final temperature	$T_0 \times 0.0001$
Probability of changing the speed	0.5
Probability of changing the time slot	0.5
Conflicts weighting coefficient, λ	1
Deviation weighting coefficient, γ	0.06
Time length of the sliding window, W	5400 s
Time shift of the sliding window, S	1200 s

TABLE V
COMPARISON OF THE TWO METHODS

Scenario		1	2	3
Initial conflicts		626	1642	1510
SA	Residual conflicts	0	0	48
	CPU time	252 s	687 s	896 s
SA+ sliding-window	Residual conflicts	0	0	16
	CPU time	191 s	347 s	469 s

TABLE VI
COMPARISON OF THE TWO METHODS FOR SCENARIO 2

Method	SA algorithm	SA+sliding-window
Number of aircraft without time-slot changes	27 (7.6%)	136 (38.3%)
Number of aircraft without speed changes	45 (12.7%)	153 (43%)
Number of aircraft without any change	16 (4.5%)	133 (37.5%)
Average delay of entrance time at TMA	86 s	81 s
Entrance delay standard deviation	177 s	160 s
Minimum entrance delay change	-100	-100
Maximum entrance delay change	495	500
Average speed change in %	0.3	0.3
Speed change standard deviation in %	5.7	4.6
Minimum speed change in %	-10	-10
Maximum speed change in %	10	10

aircraft is presumed to execute exactly one route. In future study, we intend to allow vectoring adjustment to absorb any remaining conflicts.

Table VI gives the results in more detail for scenario 2, one observes that the sliding-window approach modifies much fewer aircraft decision variables than the pure SA algorithm does, which is of practical significance. Both approaches yield similar averages and standard deviations for speed–change percentage and delay of entrance time. The results are similar for scenario 1, 2 and 3.

V. CONCLUSIONS AND FUTURE DEVELOPMENT

In this paper, we introduced a mathematical formulation of the aircraft merging problem, considering minimum separation constraints. Then, a new sliding-window approach combined with simulated annealing algorithm is proposed to solve the problem. To do so, we further introduced four different status into which the aircraft are classified depending on their relative time position within the current sub-window. This hybrid algorithm successfully reached conflict-free solutions within less computational time and less aircraft deviations than the pure SA algorithm.

Further developments may focus on tests with more scenarios, and other optimization methods to solve the same (sub-) problems. The model could also be extended to the airport, in view of optimizing the approaching procedure and the ground movements simultaneously, in order to maximize the airport throughput.

ACKNOWLEDGMENTS

This work has been supported by Civil Aviation University of China and by French National Research Agency (ANR) through JCJC program (project ATOMIC nANR 12-JS02-009-01). We would like to thank Serge Roux for his assistance with data, technical support and helpful discussions.

REFERENCES

[1] Airbus, "Global market forecast 2015-2034," Technical report, Tech. Rep., 2015.
[2] C. Zuñiga, D. Delahaye, and M. A. Piera, "Integrating and sequencing flows in terminal maneuvering area by evolutionary algorithms," in *DASC 2011, 30th IEEE/AIAA Digital Avionics Systems Conference*. IEEE, 2011, pp. 2A1–1 – 2A1–11.
[3] J. E. Beasley, M. Krishnamoorthy, Y. M. Sharaiha, and D. Abramson, "Scheduling aircraft landings - The static case," *Transportation Science*, vol. 34, no. 2, pp. 180–197, 2000.
[4] L. Bianco, P. DellOlmo, and S. Giordani, "Scheduling models for air traffic control in terminal areas," *Journal of Scheduling*, vol. 9, no. 3, pp. 223–253, 2006.
[5] J. A. Bennell, M. Mesgarpour, and C. N. Potts, "Airport runway scheduling," *4OR*, vol. 9, no. 2, pp. 115–138, 2011.
[6] H. Balakrishnan and B. Chandran, "Scheduling aircraft landings under constrained position shifting," in *AIAA Guidance, Navigation, and Control Conference and Exhibit, Keystone, CO*, 2006.
[7] X.-B. Hu and E. Di Paolo, "Binary-representation-based genetic algorithm for aircraft arrival sequencing and scheduling," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 2, pp. 301–310, 2008.
[8] F. Furini, M. P. Kidd, C. A. Persiani, and P. Toth, "Improved rolling horizon approaches to the aircraft sequencing problem," *Journal of Scheduling*, pp. 1–13, 2015.
[9] ICAO, "Air traffic management," *Doc-4444*, 2007.
[10] M. J. Frankovich, "Air traffic flow management at airports: A unified optimization approach," Ph.D. dissertation, Massachusetts Institute of Technology, 2012.